

本节内容

简单选择 排序

知识总览

选择排序

简单选择排序

堆排序

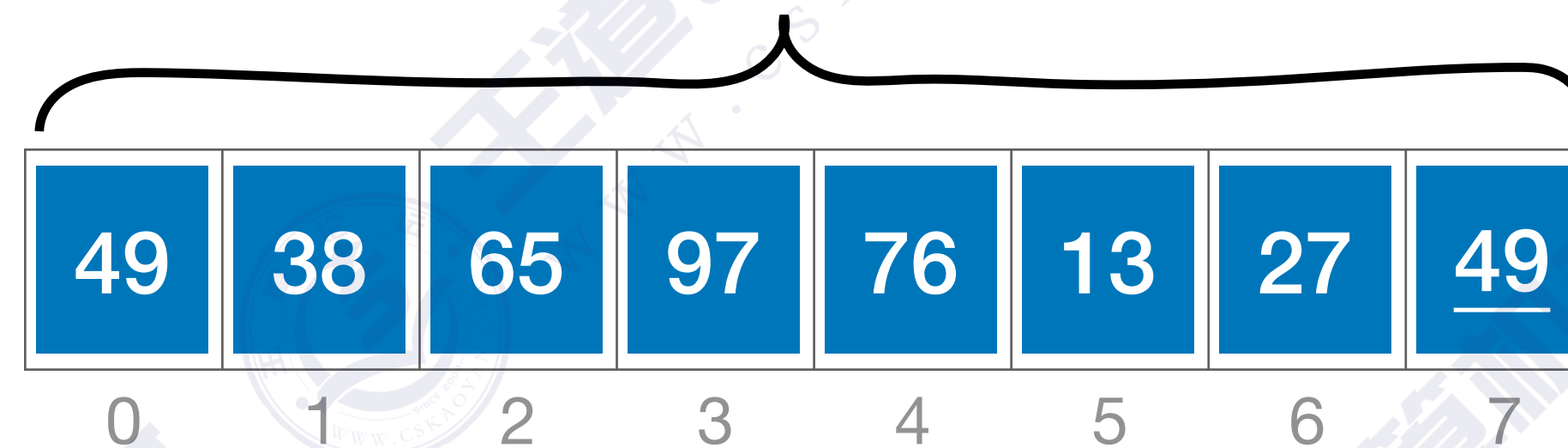
选择排序：每一趟在待排序元素中选取关键字最小（或最大）的元素加入有序子序列

简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

待排序元素

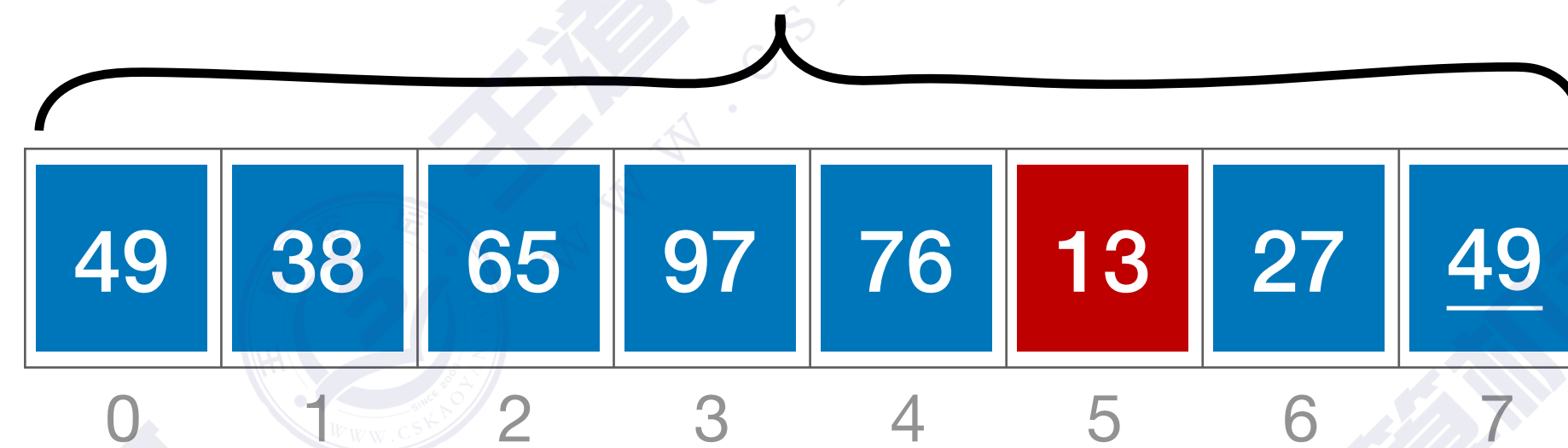


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

待排序元素

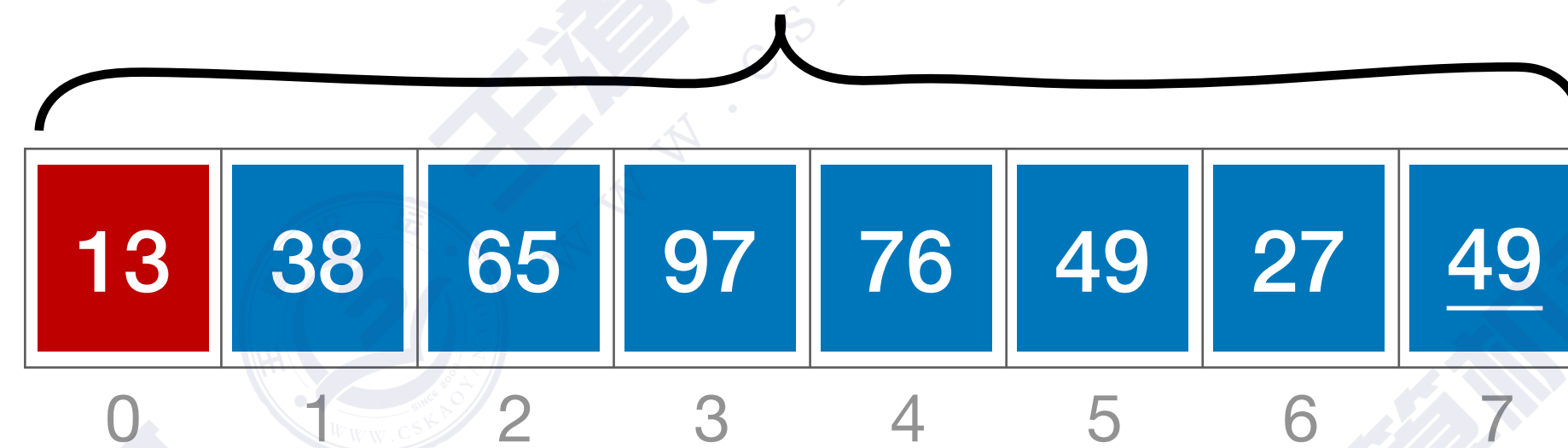


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

待排序元素



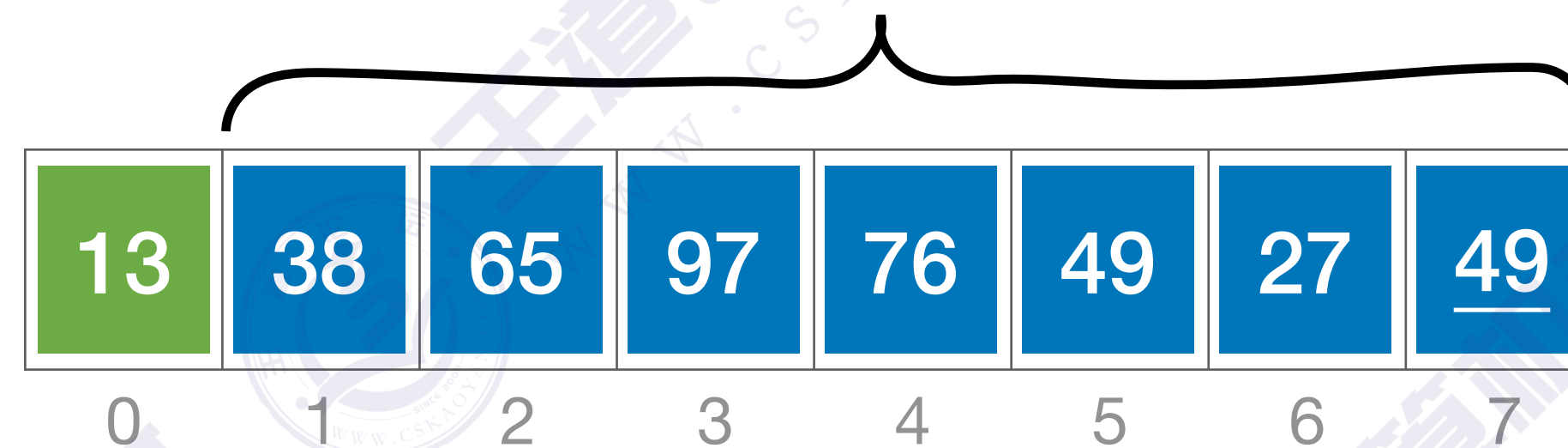
简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

待排序元素

第1趟排序结束:

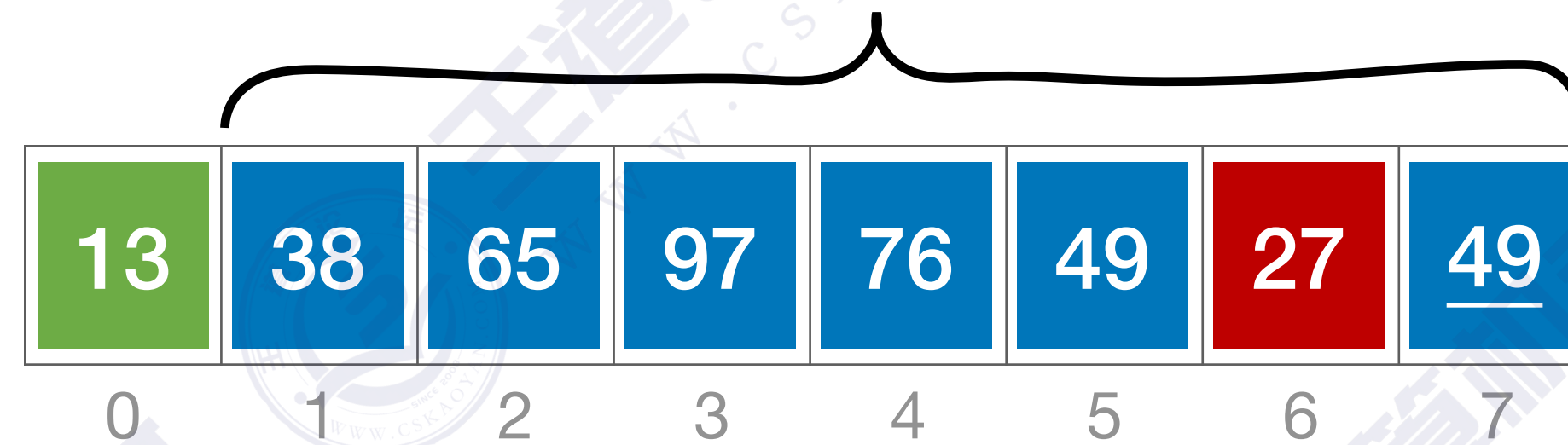


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

待排序元素

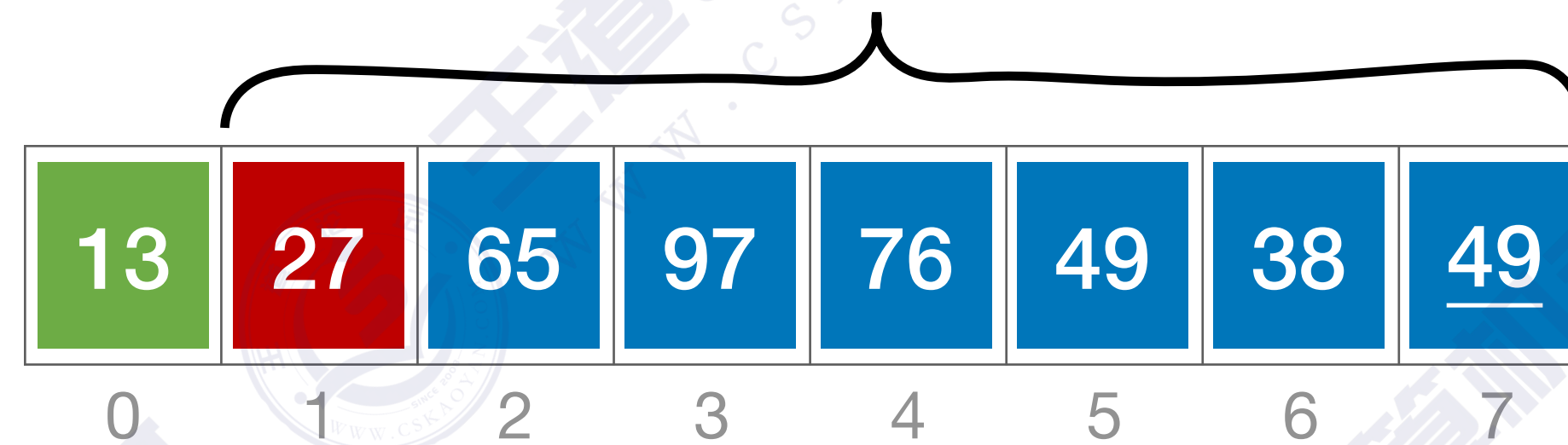


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

待排序元素

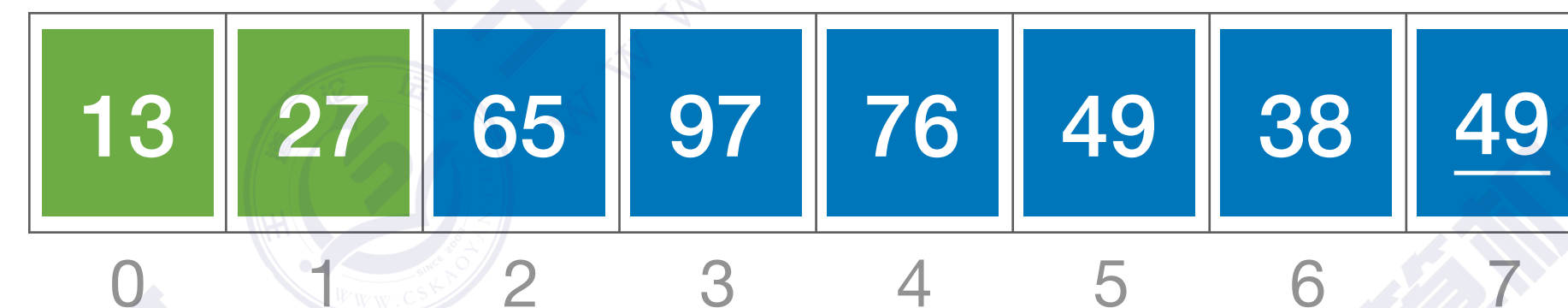


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

第2趟排序结束：

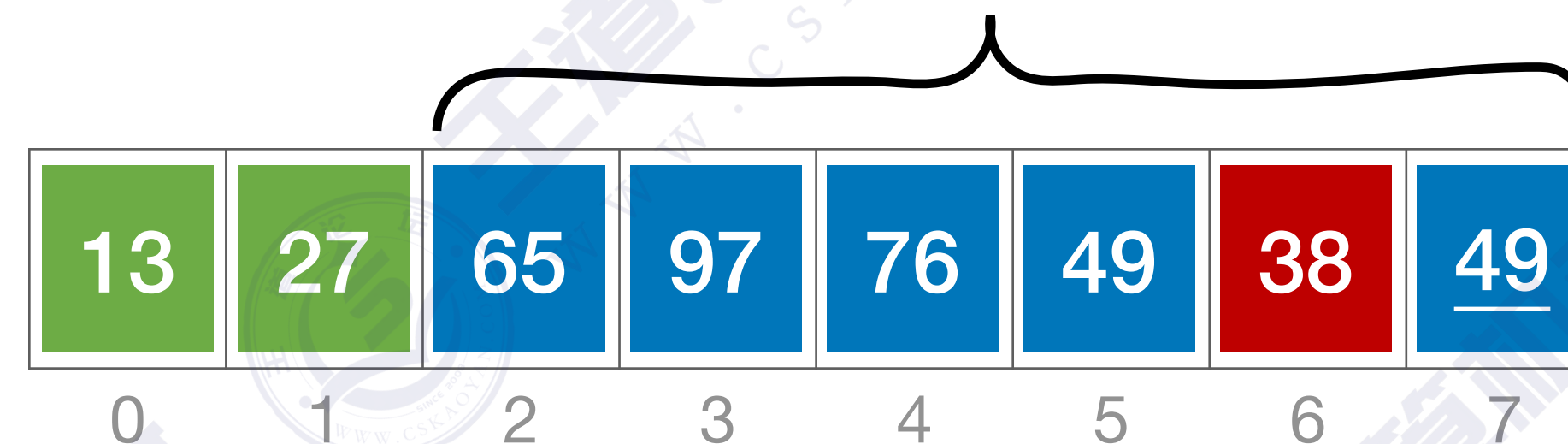


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

待排序元素

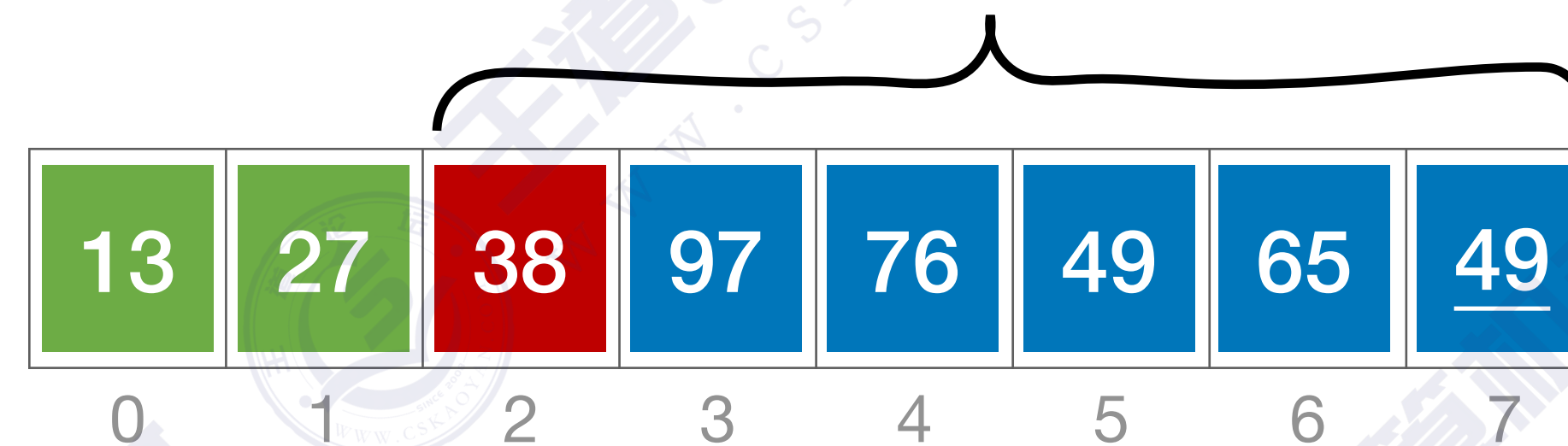


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

待排序元素

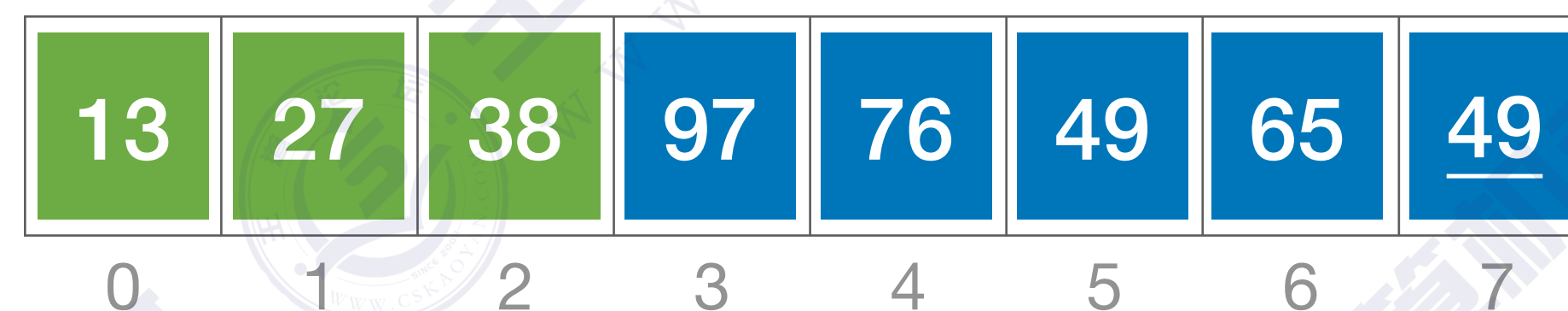


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

第3趟排序结束:

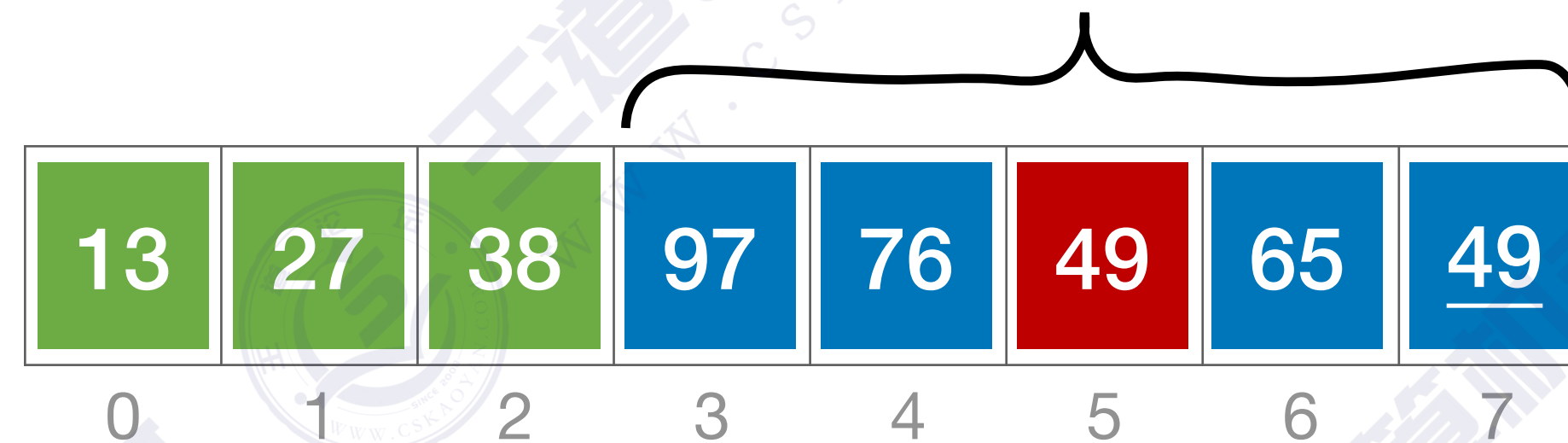


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

待排序元素

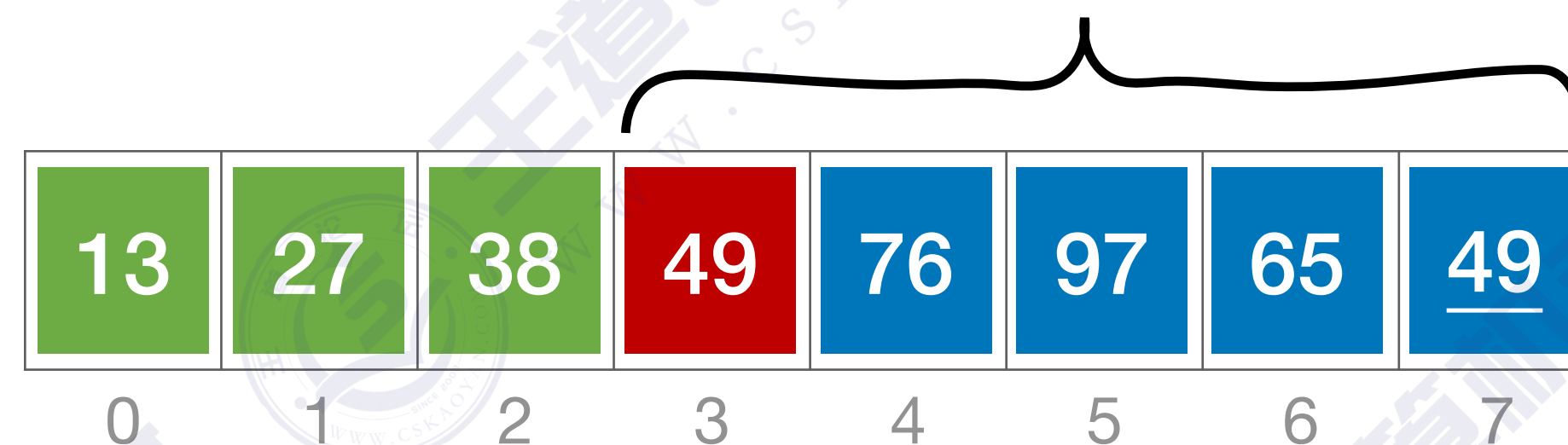


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

待排序元素

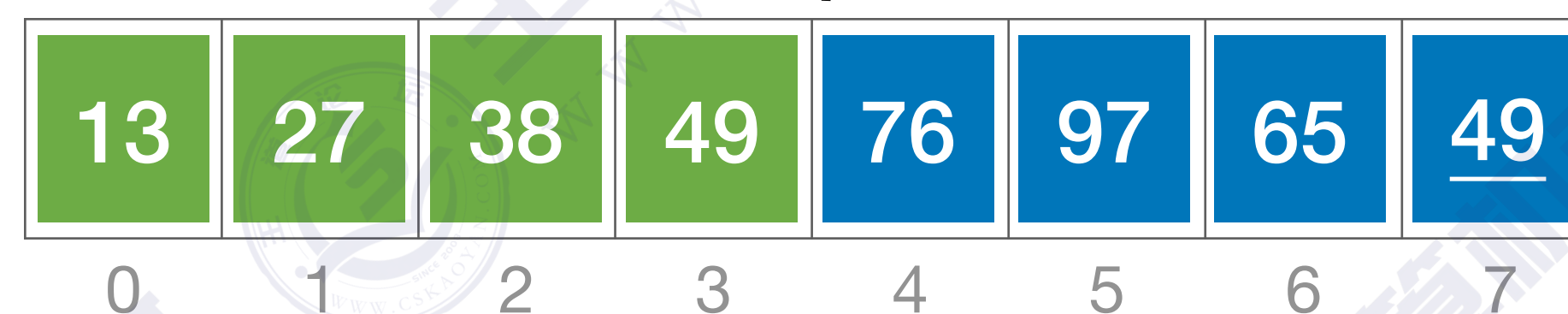


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

第4趟排序结束:

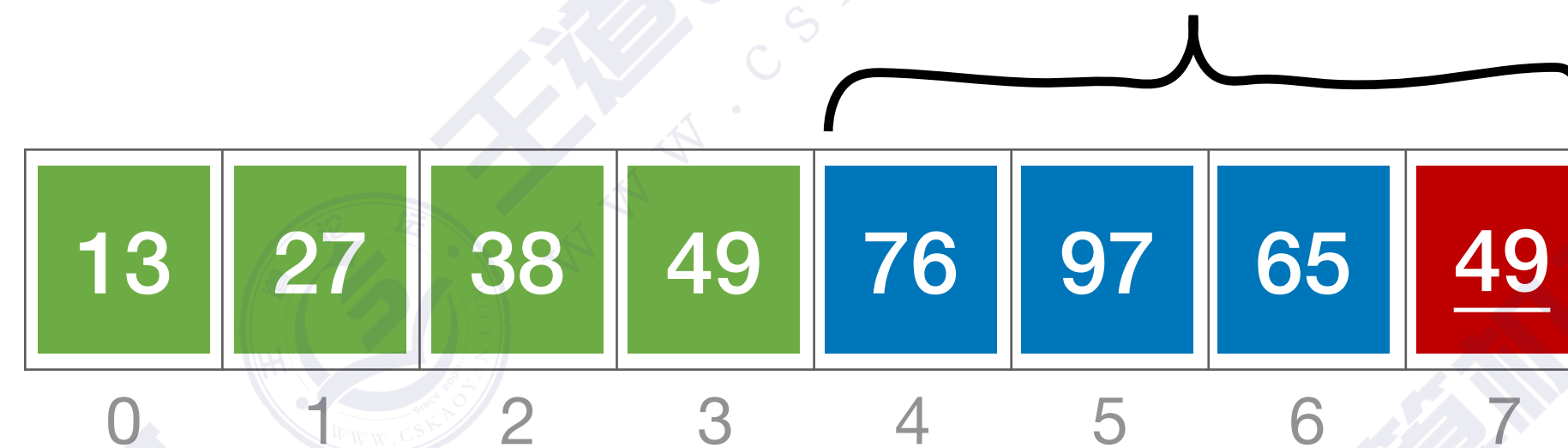


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

待排序元素



简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

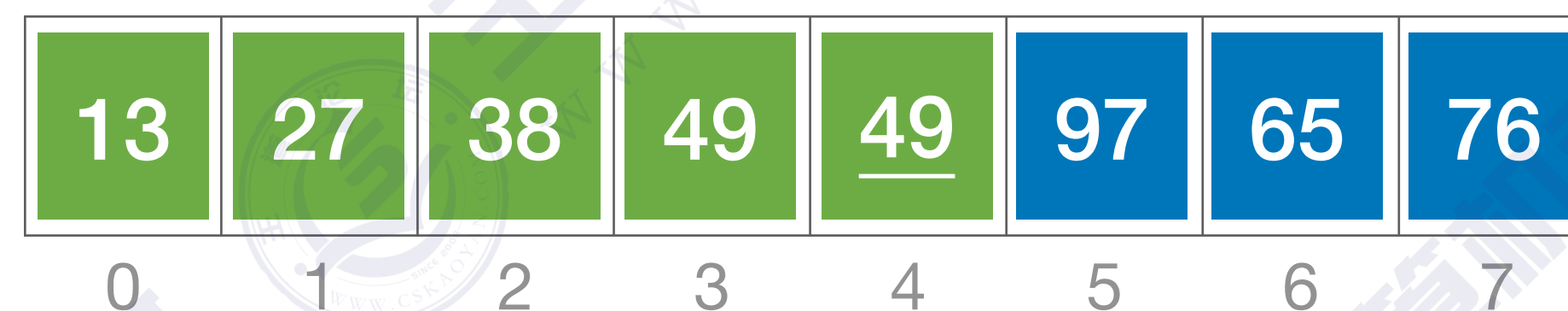


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

第5趟排序结束:



简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列



简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

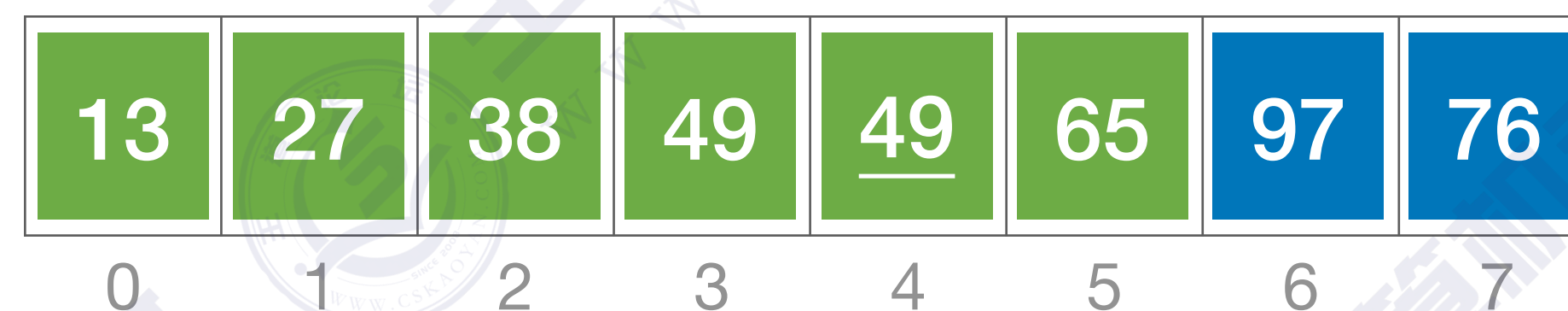


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

第6趟排序结束：



简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列



简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

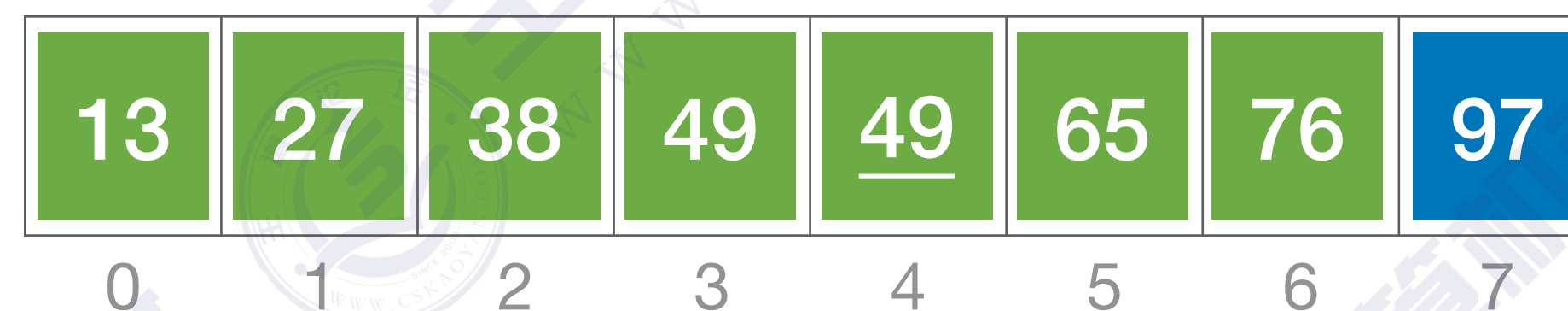


简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

第7趟排序结束:



最后剩一个不用再处理

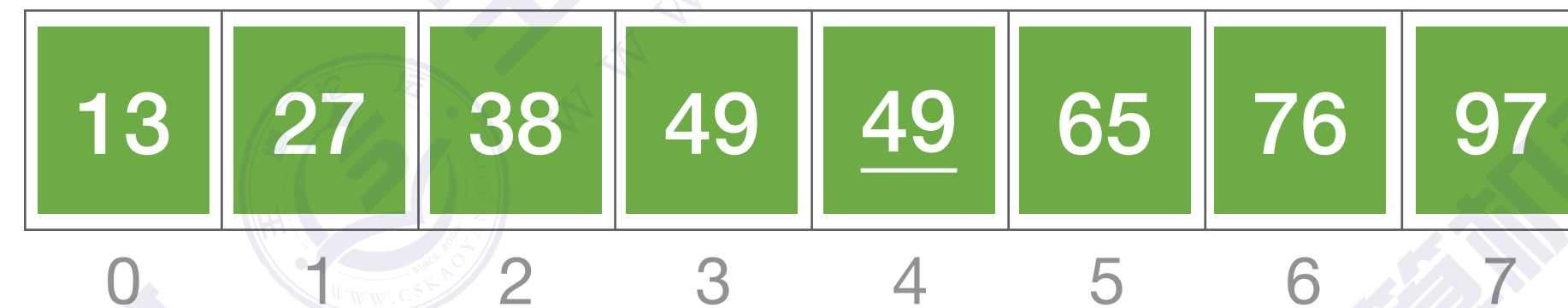
简单选择排序



每一趟在待排序元素中选取关键字最小的元素加入有序子序列

n个元素的简单选择排序需要 $n-1$ 趟处理

第7趟排序结束:



算法实现

//简单选择排序

```
void SelectSort(int A[],int n){  
    for(int i=0;i<n-1;i++){  
        int min=i;  
        for(int j=i+1;j<n;j++){  
            if(A[j]<A[min]) min=j;  
            if(min!=i) swap(A[i],A[min]);  
        }  
    }  
}
```

//一共进行 $n-1$ 趟

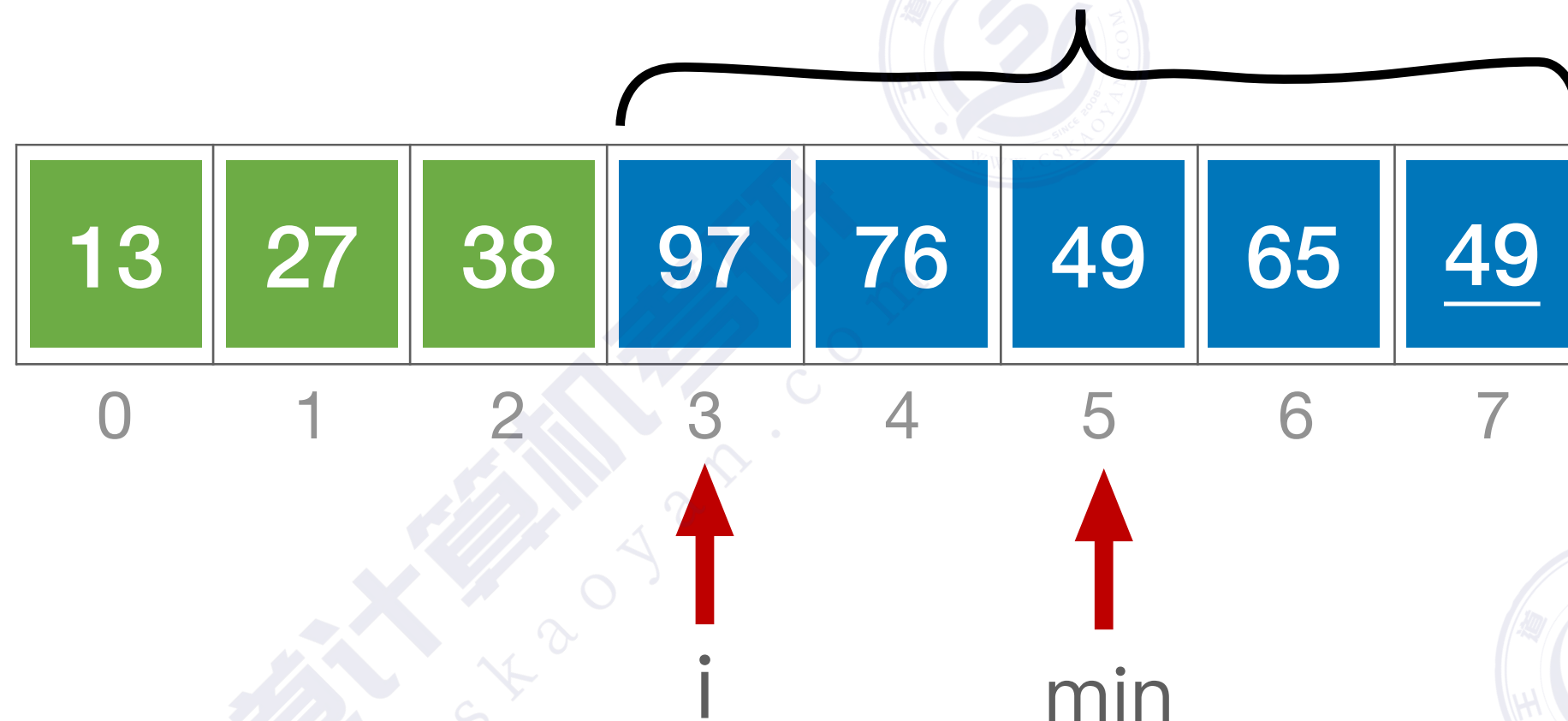
//记录最小元素位置

//在 $A[i...n-1]$ 中选择最小的元素

//更新最小元素位置

//封装的`swap()`函数共移动元素3次

待排序元素



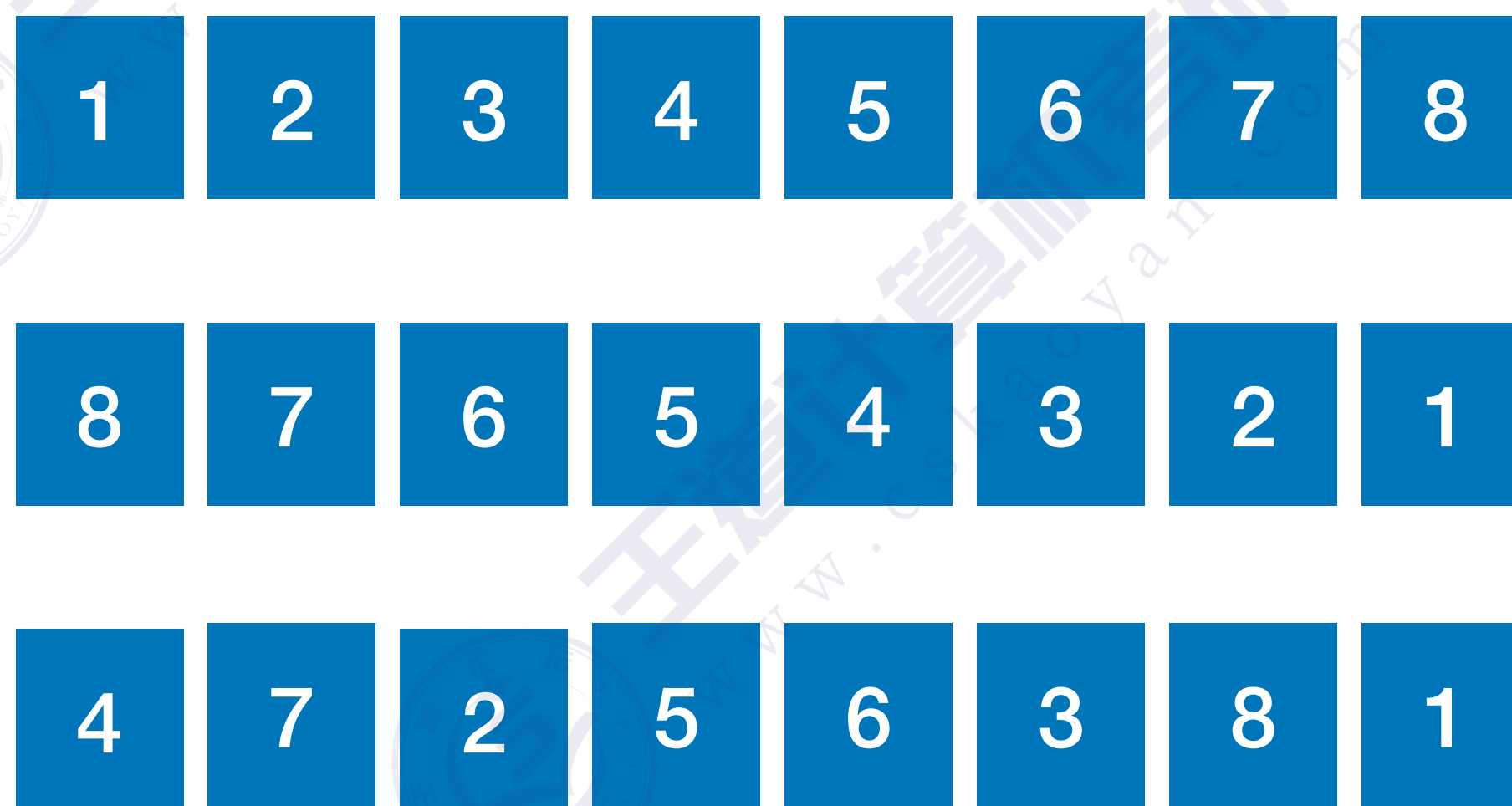
//交换

```
void swap(int &a, int &b){  
    int temp = a;  
    a = b;  
    b = temp;  
}
```

算法性能分析

空间复杂度：O(1)

时间复杂度=O(n²)



无论有序、逆序、还是乱序，一定需要 $n-1$ 趟处理

总共需要对比关键字 $(n-1)+(n-2)+\dots+1 = \frac{n(n-1)}{2}$ 次

元素交换次数 $< n-1$

算法性能分析



2 2 1

第1趟排序结束:

1 2 2

第2趟排序结束:

1 2 2

稳定性: 不稳定

适用性: 既可以用于顺序表, 也可用于链表

知识回顾与重要考点

简单选择排序

算法原理

每一趟在待排序元素中选取关键字最小的元素加入有序子序列

必须进行总共 $n-1$ 趟处理

性能

空间复杂度 $O(1)$

时间复杂度 $O(n^2)$

稳定性 不稳定

适用性 顺序表、链表都可以